Efficient Peer Selection in P2P JXTA-based Platforms

Fatos Xhafa
Dept. of Languages and Informatics Systems, UPC
Campus Nord, C/Jordi Girona 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.edu

Thanasis Daradoumis
Department of Information Sciences, UOC
Av. Tibidabo, 39-43, 08035 Barcelona, Spain
adaradoumis@uoc.edu

Leonard Barolli
Dept. of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan
barolli@fit.ac.jp

Raul Fernández, Santi Caballé
Department of Information Sciences, UOC
Av. Tibidabo, 39-43, 08035 Barcelona, Spain
{rfernandezco,scaballe}@uoc.edu

Vladi Kolici
Department of Electronics
Polytechnic University of Tirana
Mother Teresa Square, Nr.4, Tirana, Albania

Abstract

P2P systems are nowadays being used not only for file sharing but also for developing large-scale distributed applications. As an emerging paradigm for distributed computing, P2P systems are raising important issues as many novel aspects have to be dealt with in such systems. One key issue in P2P distributed computing is the efficient discovery and selection of peers, which is needed for many purposes such as efficient allocation of jobs to peers, load balancing, efficient file transfer, etc. Existing P2P distributed applications use ad hoc ways to discover and select peers, usually without any performance guarantee. In this paper we address the problem of the efficient peer selection in P2P distributed platforms. To this end, we have developed a P2P distributed platform using Sun's JXTA technology, which is endowed with resource brokerage strategies to efficiently select peers using four selection models: (a) economic scheduling model; (b) priced-based model; (c) peer-priority selection model; and, (d) random selection model. These different models are aimed to match different needs of P2P applications. Next, we have deployed the P2P JXTA platform in a real network using nodes of the PlaneLab –a planetary scale distributed infrastructure– and have experimentally evaluated the performance of the peer selection models by using a distributed application for processing large size log files of a virtual campus, which requires both efficient file transmission and processing in P2P nodes. The results of our work showed the need to develop, implement and evaluate appropriate models for efficient peer selection to match the different requirements of large-scale P2P distributed applications. Although we have used a concrete technology such as JXTA, our approach is applicable in a more general context of P2P and Grid computing domain. Finally, our approach to peer selection through brokerage services is very flexible allowing extensions with other models.

1550-445X/08 $25.00 © 2008 IEEE
DOI 10.1109/AINA.2008.10

1. Introduction and motivation

P2P systems are evolving as new a distributed computing paradigm for the development of large-scale distributed applications for solving complex problems from science and engineering by exploiting the large computing capacity offered by the nodes of the system altogether. The improvement of P2P protocols is enabling the development of P2P applications others than the well-known file-sharing applications. However, there is still few work to bring P2P system to real word applications, mainly due to the lack of robust P2P platforms that would allow the deployment of large P2P systems, in particular for efficiently discovering and selecting peers. Some advances are being done in this direction; for instance, the JXTA platform [4, 16, 17] is making possible the development of P2P real-world applications. Moreover, projects such as seti@home [20] are showing the feasibility of using P2P platforms for real life applications.

This work is motivated by the need to design and implement several models for peer selection in P2P applications. The aim is to implement and evaluate these models independently of P2P application domains in a way that they could serve to the development of high performance P2P application in general, that is, to facilitate the use of P2P infrastructures as distributed computing environments. The need for efficient peer selection arises in many P2P applications such as for job allocation, fast file transfer, etc. Therefore, no one model would be able to match the requirements of different scenarios/applications; rather, several models must be studied in order to identify which of them works best under which P2P infrastructure and/or job characteristics. The peer selection models considered in this work range from a simple random model to more advanced economic-based models. These models are the following. (a) *Economic scheduling model* [7]: in this model the idea is to find/provision as many as possible available idle peers to which the new incoming jobs can be allocated. Crucial to this model is the ready time of peers in order to plan in advance the allocation of jobs to P2P nodes. (b) *Priced-based model* (e.g. [22]): in this model peers are associated a cost, which is computed using different criteria that range from peer's state to P2P infrastructure parameters. (c) *Peer-priority selection model*: in this model it is the user who selects the peer, among different candidate peers, based on previous traces/experiences of job allocations submitted by the user. And, (d) *random selection model*: this is the simplest model in which a peer is selected uniformly at random among several peer candidates. It should be noted that the use of economic models in P2P systems is a hot research topic nowadays [5, 9, 21, 13, 18].

Our approach is exemplified using the Sun's JXTA open protocols and has been validated using a simple distributed application scenario. We have joined the PlanetLab platform [14] –a planetary scale distributed infrastructure– and used a slice of nodes to deploy a P2P network and have experimentally evaluated the performance of the proposed peer selection models. A distributed application for processing large size log files of a virtual campus, which requires both efficient file transmission and processing was chosen for the experimental evaluation.

The rest of the paper is organized as follows. We briefly describe some related work in Section 2. The architecture of the P2P platform is presented in Section 3. In Section 3.2 we give the peer selection models considered in this work. The evaluation of the proposed models is given in Section 4. Finally, we conclude in Section 5 with some remarks and indicate directions for future work.

2. Related work

P2P systems are novel in both technological aspects and design/implementation issues. Recently, considerable research effort is being done on several important issues related P2P systems. Much of this effort has been addressed on overlay networks [2, 3, 1] and quite a few address the design and implementation of libraries to support the development of distributed applications. Also, the issue of discovery, resource location and allocation is addressed in several recent works [12, 11]. Buyya et al. [6] addressed issues for P2P systems by putting special emphasis on: (a) deploying internet services by overlaying; (b) the need for scalability of P2P applications that would require keeping knowledge of a small fraction of global state in each peer; and, (c) the need for load balancing, this should be separated from the P2P applica-

tions. Regarding the efficient allocation of tasks to computational resources, most of the ideas from the Grid computing domain are also applicable to P2P domain, although some differences related to existing policies on resources should be taken into account. Given that P2P networks are usually large or very large as they are based on contributions of individuals, the peer selection model should be able to find/provision as many idle peers as possible while allocating tasks to P2P nodes. On the other hand, because P2P resources belong to different individuals and/or institutions around the world, the peer selection models based on economic-like models are quite desirable for P2P systems since they allow to easily incorporate incentive mechanisms, which are important for the deployment of P2P systems. One such interesting selection model is the one proposed by Ernemann et al. [7] for economic scheduling in Grid computing. Other related approaches are by Yu et al. [22] and Ping et al. [18]; in this later work JXTA technology is used.

3. The architecture of our P2P platform

In this section we present the architecture of the P2P distributed platform[1], called JXTA-OVERLAY [**?**], we have developed using JXTA technology. The main building blocks of the platform are: (a) the *Broker* module; (b) the *Primitives* module; and, (c) the *Client* module. Altogether this three modules form a new *overlay* on top of JXTA (see Fig. 1).

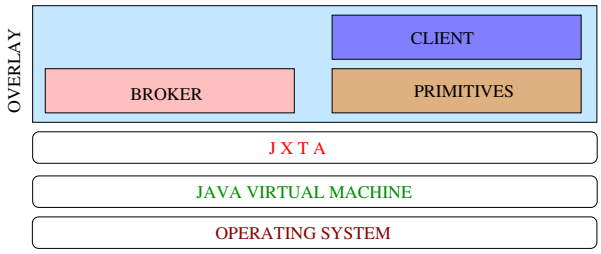

Figure 1. The Architecture of the P2P Overlay.

Importantly, the new overlay is designed and implemented to be totally independent of any possible P2P applications, which will be built on top of the overlay. Clearly, one of the characteristics of the primitives module (see below) is their independence from the applications that will be using them. We give next a basic description of the three modules of the overlay.

Primitives: The objective of the overlay is to provide a set of basic functionalities, that we call primitives since they will be part of any P2P application, as regards the discovery and allocations of resources. This set of primitives is intended to be as complete as possible as regards the functionalities for the discovery and allocations of resources. Roughly speaking, the set of primitives includes functionalities that allow: peer discovery, peer's resources discovery, peer selection, resource allocation, file/data sharing, discovery and transmission, instant communication, peer group functionalities. The primitives are designed to be generic in a way that any application built on top of the overlay can use it as a *"black box"*. To this end, we observed that the overlay should include, apart from primitive functionalities, two other modules: a *broker layer* and a *client layer*.

Broker layer: This layer is in charge of achieving the resource allocation functionalities, resource monitoring, management of executable tasks defined in the set of primitives. Note that the broker peers do not interact with final user applications therefore they represent just one layer.

Client layer: This layer is in charge of receiving and managing all events produced in any application built on top of the overlay due to calls to the primitives.

By using the above architecture, we achieved the set of primitives to be completely independent of any application as the client layer will allow (final) user applications to communicate with the overlay. Moreover, the primitives allow to keep the intrinsic decentralized nature of Grid/P2P systems. The idea of the architecture using brokers has been initially explored in [19]. The set of primitives that allow to accomplish the aforementioned functionalities is organized in interfaces according to an affinity criterion. Thus we have the interfaces *authentication*, *resource discovery and information*, *management of executable tasks*, *file sharing, discovery and transmission*, *resource statistics*, among others. An important place in the primitives is given to functionalities related to the management of executable tasks. These functionalities are intended to give service to users/applications on top of the overlay that submit executable tasks and receive

[1]For more details and updated information on this platform, please refer to http://jxta-overlay.dev.java.net

results in turn. It should also be mentioned that the file sharing and transmission functionalities extend existing JXTA functionalities of sharing in P2P systems since an efficient file transmission is necessary for submitting tasks to resources. Resource statistics is another important interface in the overlay, and it is particularly useful for the selection of peers (statistics about the peers, the peergroups, the brokers and the clients.)

3.1. Peers, brokers and discovery using JXTA

Now we show how is implemented the set of the primitives. We take advantage that JXTA allows different types of peers in order to classify them into two groups: *client peers* and *broker peers*. The former are *complete edge peer* while the latter act as *rendezvous* and *relays*.

Broker peers. Brokers are the *governors* of the network: they are connected to the to P2P platform and are in charge of receiving and allocating the requests sent by clients of the peerGroup. Whenever a broker receives a request, it selects, according to one or more peer selection models, the best peer candidate for processing that request and makes the allocation. It should be noted that the definition of the broker peers allows to keep the control on the resource allocation. Thus, any peerGroup has (at least) a broker to which client peers get connected and send their resource allocation petitions. This is done by redefining the JXTA rendezvous, the Pipe, Discovery and Rendezvous of JXTA (denoted RendezvousOV –rendezvous overlay–, PipeOV, etc.). This is mainly done to ensure reliability of the overlay. Among broker's functionalities we distinguish: (a) event management; (b) controlling the resources connected to the broker; (c) maintaining the organization of resources in groups; (d) find the best resource for the file sharing; (e) find the best resource (according to scheduling policy/economic models) for task execution; and, (f) maintain updated statistic information (as regards task executions, file transfers, etc.). Further, we also note that the design of the broker is organized in several layers/modules: *brokerCore*, *brokerManager* and *brokerFunctions* (we omit the details.)

Client peers. These are peers that instantiate the Client module, which serves as a communication layer between the primitives and the final user application. A client peer is in charge of receiving and managing all events produced in any application (built on top of the overlay) due to calls to the primitives. It is organized in a similar way as the broker: *clientCore*, *clientManager* and *clientFunctions*.

3.2. Peer selection models

As part of the set of primitives we have implemented four models for peer selection. These primitives are then used as resource brokerage strategies by the broker peers. The peer selection models considered in this work range from a simple random model to more advanced economic-based models. These models are: *Economic scheduling model*; (b) *Priced-based model*; (c) *Peer-priority selection model*; and, (d) *random selection model*.

Economic scheduling model. In this model [7] the idea is to find/provision as many as possible available idle peers to which the new incoming jobs can be allocated. Crucial to this model is the ready time of peers in order to plan in advance the allocation of jobs to P2P nodes. Thus, many parts of the problem are processed in parallel in different peers and peers can communicate among them during task realization. In this model is crucial the expected starting time to compute of a given peer for a given task. In the case of task execution this information is either extracted from historical data or is specified by the user[2]. On the other hand the peer advertisements are very important to know the state information of peers. However, this could be problematic for tasks needing a short or very short execution time since advertisement are periodically updated. In this case, and estimated time is computed by the broker based on historical data kept for the peers. In case several peers are available candidates for executing the task, some additional criteria such as CPU speed are used.

Priced-based model. In this model[3] peers are associated a cost, which is computed using different criteria that range from peer's state to P2P infrastructure parameters. The set of criteria used to identify the best peer(s) are classified into: (a) *global criteria* (percentage of successfully sent messages in the current ses-

[2]The reader is also referred to [10] (The Cornell Theory Center) and the Parallel Workload Archive [15].

[3]Also referred to as Data model.

sion, percentage of successfully sent messages in all sessions (total), percentage of successfully sent messages during the last k-hours; number of messages in the outbox queue now, average number of messages in the outbox queue, number of messages in the inbox queue now, average number of messages in the inbox queue, average number of attempts in outbox in the current session, average number of attempts in outbox in all sessions (total), etc.; Amount of sent bytes in the current session, similarly for all sessions (total); Peer's bandwidth IN, Peer's bandwidth OUT, etc. (b) *specific task execution criteria*[4] (percentage of successfully executed tasks in the current session, percentage of successfully executed tasks in all sessions (total), percentage of tasks accepted by the peer for execution in the current session, percentage of tasks accepted by the peer for execution in all sessions (total), percentage of cancelled tasks in the current session, percentage of cancelled tasks in all sessions (total), etc.. (c) *specific file request criteria*[5] (percentage of sent files in this session, percentage of sent files in all sessions (total), percentage of cancelled file transfers in the current session; Average file transfer ratio, etc.

Each of the above criteria is given a certain weight (either user defined or pre-specified) meaning that some criteria are more important than others or even some are negligible (of zero weight). The broker, upon receiving a request (task execution or file transfer) from a peer, evaluates the above criteria, applies the weights and thus assigns a *price* (a score) to each candidate peer. The best score peer is then chosen for executing the task. The user can specify two ways of computing the peer's score, namely *fixed point* (w.r.t. absolute position in the peer list) and *variable point* (w.r.t relative position in the peer list). Regarding the criteria's weights, the following specific ways have also been implemented: *all disabled* (no weights are considered, the peer is randomly chosen); *same priority* (the weights are the same, i.e., criteria are equally important); *quickest peer* (only the criteria related to task execution performance and file transmission are considered independently of the peer reliability); *reliable peer* (only criteria related to peer reliability w.r.t. task execution/file transmission are considered, independently of peer's performance); *balancing peer* (only the criteria related to load balancing are considered).

Peer-priority selection model. In this model it is the user who selects the peer, among different candidate peers based on previous traces/experiences of request (task execution or file transmission) submitted by the user. This model is useful when the user knows the performance of some peers in advance, for instance, from previous submissions of the tasks. In this case, the broker has to just assure that the selected peer is available for executing the task and therefore this model has a very low computational cost as opposed to the previous models.

Random selection model. This is the simplest model in which a peer is selected uniformly at random among several peer candidates. Although simple, this model could be useful when peer candidates are almost homogeneous.

We show in Fig. 2 the UML diagram of the proposed models instantiated by the broker module.

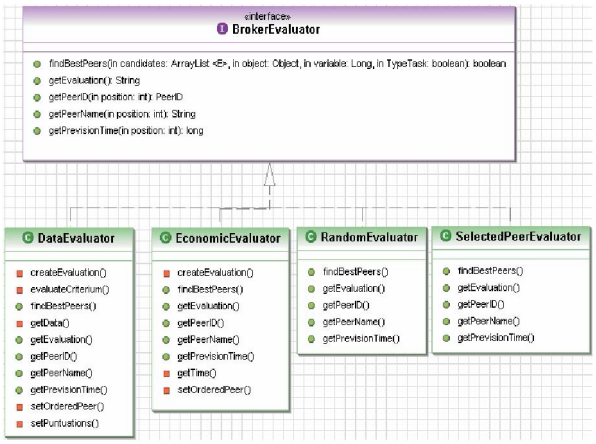

Figure 2. Diagram of peer selection models.

4. Experimental evaluation

4.1. Deployment of the P2P network

In order to evaluate the performance of the presented peer selection models, first we deployed the P2P network using nodes of the PlanetLab platform.

[4] A total of 47 criteria has been implemented in this model for peer selection for task execution.

[5] A total of 39 criteria has been implemented in this model for peer selection for file transmission

PlanetLab [14] is an open platform for developing, deploying and accessing planetary-scale services. It is, at the time of this writing, composed of 782 nodes at 382 sites. Each Planetlab node is an IA32 machine that must comply with minimum hardware requirements (i.e. 1GHz PIII + 1Gb RAM) running the same base software, basically a modified Linux operating system offering services to create virtual isolated partitions in the node, called slivers, which look to users as the real machine. Planetlab allows every user to dynamically create up to one sliver in every node, the set of slivers assigned to a user form what is called a slice. It is said that a Planetlab node can run up to 100 concurrent slivers. The sample of PlanetLab's machines forming our slice is about 25 nodes. Moreover we used the cluster nozomi.lsi.upc.edu (a main control node + five computing nodes). The main node was used as one the brokers of the P2P network.

4.2. The distributed application scenario

Next, we have chosen a simple but representative application to run on the resulting P2P platform. This application consists in processing large log files kept by the Virtual Campus at the Open University of Catalonia (http://www.uoc.edu), which offers distance education through the Internet in different languages. As of this writing, about 40,000 students, lectures and tutors from everywhere participate in some of the 23 official degrees and other PhD and post-graduate programs resulting in more than 600 official courses.

All users' requests are chiefly processed by a collection of Apache [10] web servers. Each web server stores in a log file all users' requests received in this specific server and the information generated from processing the requests. Once a day , all web servers in a daily rotation merge their logs producing a single very large log file containing the whole user interaction with the campus performed in the last 24 hours. A typical daily log file size may be up to 10 GB. Log file entries are structured following a type of format known as Common Log Format [8]. Unfortunately the log file is not human readable making thus indispensable its processing to extract relevant information that would serve as basis for later statistical processing. The problem of processing log files of the virtual campus represent several interesting characteristics. Log files are of large size making thus relevant a parallel processing using the P2P network. Further, due to their structure (Common Log Format) the log file can be very easily parallelized using the Master-Worker paradigm since the file can be split by a master node into many independent parts and processed in parallel by other peer nodes (slaves). Finally, the processing requires efficient file transmission.

4.3. Computational results and evaluation

For the experimental study we used daily log files and well-stratified short samples of about 100Mb consisting of representative daily periods with different activity degrees (e.g. from 7 p.m. to 1 a.m. as the most active lecturing period). The computational results presented here are obtained by running the same experiment five times and the results are averaged.

We show in Fig. 3 the time needed by the broker to find the best candidate for each model when the log file was split into 1, 4 and 16 parts. As can be seen from this figure, the data model (price-based model) is the most computationally expensive among the proposed models.

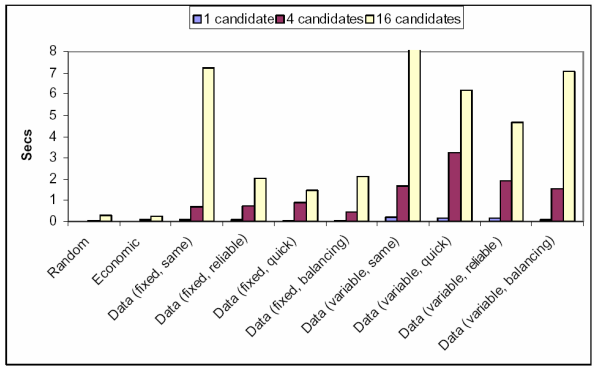

Figure 3. Broker's processing time for discovering the best peer.

Two different modes for sending files to peers for processing were used: the ftp transfer (that is, peers download the file chunks from an FTP site) and JXTA file transfer. We show in Fig. 5 the resulting processing time of log files of 100MB when using the best peer found according to economic model and price-

based model[6] (the most relevant for this experimental study.) Further, in Fig. 4 the processing time without taking into account the file transmission time (from the master node to peers and vice-versa) is shown.

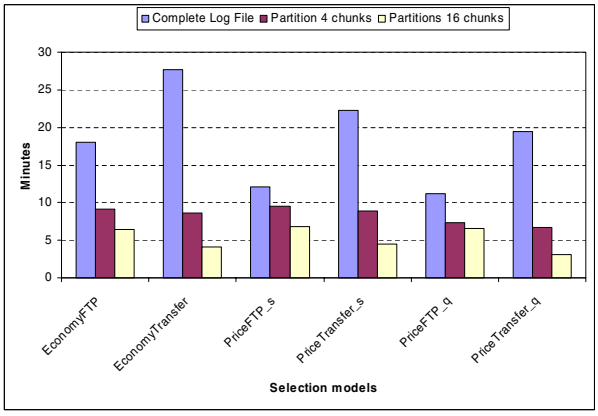

Figure 4. Total processing time of log files in the P2P platform.

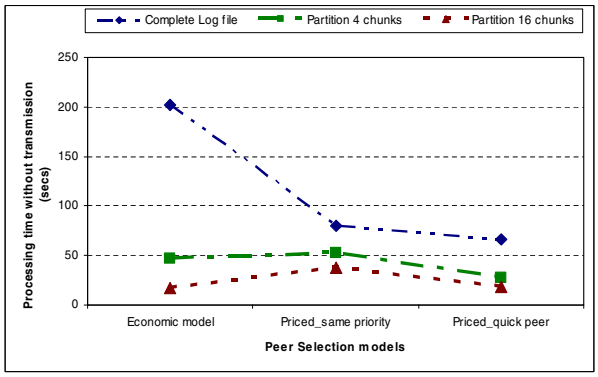

Figure 5. Processing time of log files in the P2P platform (without transmission time).

As can be seen from the above results, as expected, it's worth using the P2P network to process the log files. In particular sending just one file via FTP takes most of the overall processing time while it is much more efficient to split the file into chunks and send them at the same time to different peers, achieving thus different degrees of granularity. Then, when partitioning the file into chunks, the direct JXTA transfer seems to perform better than the FTP transfer. We noticed however that the file transmission was the most time consuming overall. Regarding the different selection models, they showed different performance. The price-based model with quick peer, which computes the best peer w.r.t. the peer's communication and peer's historical performance showed to perform better. On the other hand, the price-based model with the same priority performed not as good and showed a higher computational cost.

It should be noted however that the performance of different models depends on the state of the network; in particular the economic model could perform better if the provision of task allocation is relevant. To see this effect, we considered the following simple scenario: the log file was split into four chunks and 8 peers were candidates for processing them. The 4 chunks were submitted for processing twice. The results showed now to be different: the economic model used the four best peers for processing the 4 chunks and then used exactly the same peers for processing the second battery of 4 chunks while the economic model sent the 4 chunks to the four best peers and next sent the second battery of four chunks to the four idle peers.

5. Conclusions and future work

In this work we have presented four models (economic scheduling model, priced-based model, peer-priority selection model and random selection model) for peer selection in a P2P JXTA-based platform. These models are implemented and a first evaluation is done in a real P2P network that uses, among others, nodes of the PlanetLab platform. The performance of the proposed models is studied by using a distributed application scenario for processing large size log files of a virtual campus.

In our future work we would like to measure the performance of the proposed peer selection models in large scale distributed application involving a large number of peers as well as a large number of tasks to be allocated to the peer nodes. Also, we plan to investigate other peer selection models and extend the experimental results of this study.

[6]The notation in the figure reads as follows: EconomyFTP: economic model using ftp; EconomyTransfer: economic model using Jxta transfer; PriceFTP_s: Price-based model with same priority using ftp; PriceTransfer_s: Price-based model with same priority using Jxta transfer; PriceFTP_q: Price-based model with quick peer using ftp; PriceTransfer_q: Price-based model with quick peer using Jxta transfer.

Acknowledgements

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

References

[1] L. Alima, A. Ghodsi, and S. Haridi. A framework for structured peer-to-peer overlay networks. In *Global Computing*, 223–249, 2004.

[2] D. Andersen, H.. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. of the 18th ACM Symposium on Operating Systems Principles*, Canada, 2001.

[3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Experience with an evolving overlay network testbed. *ACM SIGCOMM Computer Communication Review*, 33(3):13–19, 2003.

[4] D. Brookshier, D. Govoni, N. Krishnan, and J. Soto. *JXTA: Java P2P Programming*. Sams Pub., 2002.

[5] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[6] J. Crowcroft, T. Moreton, I. Pratt, and A. Twigg. Peer-to-Peer Technologies. In Foster and Kesselman, eds, *The Grid: Blueprint for a New Computing Infrastructure*, chapter 29, 593–622. Morgan Kaufmann, 2003.

[7] C. Ernemann, V. Hamscher, and R. Yahyapour. Economic scheduling in grid computing. In *The 8th Int. Workshop on Job Scheduling Strategies for Parallel Processing*, 128–152, UK, 2002.

[8] Common Log Format. (As of February 2007) http://httpd.apache.org/docs/1.3/logs.html#common.

[9] C. Grothoff. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik*, (3):285–292, 2003.

[10] S. Hotovy. Workload evolution on the Cornell theory center ibm sp2. In *Proc. of Job Scheduling Strategies for Parallel Proc. Workshop,*, 27–40, 1996.

[11] H. Hsiao, M. Baker, and Ch. King. A peer-to-peer mechanism for resource location and allocation over the grid. In *ISPA04*, 604–614, 2004.

[12] H. Hsiao and Ch. King. Similarity discovery in structured P2P overlays. In *ICPP03*, 636–, 2003.

[13] J. Hwang, Ch. Lee, J. Song, and K. Pyo. Grid and p2p economics and market models. In *Proc. of the 1st Int. Workshop on Grid Economics and Business Models*, 3–18. IEEE Computer, 2004.

[14] Planet Lab. http://planet-lab.org/.

[15] Parallel workload archive. The Hebrew University Parallel Systems Lab. (As of February 2007.) http://www.cs.huji.ac.il/labs/parallel/workload/

[16] S. Li. *Early Adopter JXTA*. Wrox Press, 2003.

[17] S. Oaks, B. Traversat, and L. Gong. *JXTA in a Nutshell*. O'Reilly, 2003.

[18] T. Ping, G. Sodhy, Ch. Yong, F. Haron, and R. Buyya. A market-based scheduler for jxta-based p2p computing system. In *Int. Conf. Computational Science and Its Applications, Proc., Part IV*, vol. 3046 of *LNCS*. Springer, 2004.

[19] J.E. Riasol and F. Xhafa. Juxta-cat: a jxta-based platform for distributed computing. In *Proc. of the 4th Int. Symp. on Principles and practice of programming in Java*, 72–81, 2006. ACM Press.

[20] Seti@Home. http://setiathome.berkeley.edu/.

[21] O.4 Wolfson, B. Xu, and A. Sistla. An economic model for resource exchange in mobile peer to peer networks. *Proc.of the 16th Int. Conference on Scientific and Statistical Database Management*, 235–244. IEEE Computer Soc., 2004.

[22] J. Yu, M. Li, Y. Li, F. Hong, and M. Gao. A framework for price-based resource allocation on the grid. *Proc. of Parallel and Distributed Computing: Applications and Technologies*, vol. 3320 of *LNCS*, 341–344. Springer, 2004.